AUTOMATIC GENERATION OF REWRITE RULES FOR URLS

RELATED APPLICATIONS

[0001] This application claims priority under 35 U.S.C. § 119 based on U.S. Provisional Application No. 60/524,889, filed November 26, 2003, the disclosure of which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

A. Field of the Invention

[0002] The present invention relates generally to content retrieval on the world wide web, and more particularly, to automated web crawling.

B. Description of the Related Art

[0003] The World Wide Web ("web") contains a vast amount of information. Search engines assist users in locating desired portions of this information by cataloging web pages. Typically, in response to a user's request, the search engine returns references to documents relevant to the request.

[0004] Search engines may base their determination of the user's interest on search terms (called a search query) entered by the user. The goal of the search engine is to identify links to high quality relevant results based on the search query. Typically, the search engine accomplishes this by matching the terms in the search query to a corpus of pre-stored web documents. Web documents that

contain the user's search terms are considered "hits" and are returned to the user.

[0005] The corpus of pre-stored web documents may be stored by the search engine as an index of terms found in the web pages. Documents that are to be added to the index may be automatically located by a program, sometimes referred to as a "spider," that automatically traverses ("crawls") web documents based on the uniform resource locators (URLs) contained in the web documents. Thus, for example, a spider program may, starting at a given web page, download the page, index the page, and gather all the URLs present in the page. The spider program may then repeat this process for the web pages referred to by the URLs. In this way, the spider program "crawls" the world wide web based on its link structure.

[0006] Some web sites generate pages dynamically based on parameters specified in the URLs. Some of these parameters may be redundant and may not affect the content returned by the web site. This can result in many different URLs that all refer to the same content. Different URLs that refer to the same content can cause the spider program to redundantly index the web page. This is undesirable, as it can, for example, lead to a decrease in the quality of the operation of the search engine, can affect the feasibility of crawling particular sites, and can significantly increase the cost and time involved in crawling.

[0007] Thus, there is a need in the art to more effectively crawl web sites that use different URLs that refer to the same content.

SUMMARY OF THE INVENTION

[0008] One aspect consistent with the principles of the invention is a method that includes receiving a first uniform resource locator (URL) including one or more parameters and retrieving content corresponding to the first URL. The method further includes retrieving content corresponding to URLs having different parameter combinations of the one or more parameters, identifying the parameter combination from the URLs that corresponds to content that is approximately the same as the content corresponding to the first URL and that contains a reduced number of parameters, and generating one or more URL rewrite rules based on the identified parameter combination.

[0009] Another aspect is directed to a method for converting a URL into a canonical form of the URL. The method includes receiving a URL that refers to content and that contains a parameter set including at least one parameter. The method further includes applying a predetermined rewrite rule to the URL that removes the at least one parameter from the URL when the at least one parameter does not affect the content referred to by the URL, and outputting the rewritten URL as the canonical form of the URL.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, explain the invention. In the drawings,

[0011] Fig. 1 is an exemplary diagram of a network in which systems and methods consistent with the principles of the invention may be implemented;

[0012] Fig. 2 is a diagram of an exemplary client or server device shown in Fig. 1;

[0013] Fig. 3 is an exemplary functional block diagram illustrating an implementation of the server software shown in Fig. 1;

[0014] Fig. 4 is a diagram illustrating an exemplary URL;

[0015] Fig. 5 is a flow chart illustrating generation of rewrite rules; and

[0016] Fig. 6 is a diagram illustrating exemplary URLs.

DETAILED DESCRIPTION

[0017] The following detailed description of the invention refers to the accompanying drawings. The detailed description does not limit the invention.

[0018] As described herein, rewrite rules are generated that describe how URLs can be rewritten to reduce or eliminate different URLs that redundantly refer to the same, or substantially the same, content. The rewrite rules can be applied to URLs received when crawling a network to increase the efficiency of the crawl and the corresponding document index generated from the crawl.

EXEMPLARY NETWORK OVERVIEW

[0019] Fig. 1 is an exemplary diagram of a network 100 in which systems and methods consistent with the principles of the invention may be implemented. Network 100 may include multiple clients 110 connected to one or more servers

120 via a network 140. Network 140 may include a local area network (LAN), a wide area network (WAN), a telephone network, such as the Public Switched Telephone Network (PSTN), an intranet, the Internet, or a combination of networks. Two clients 110 and a server 120 have been illustrated as connected to network 140 for simplicity. In practice, there may be more or fewer clients and servers. Also, in some instances, a client may perform the functions of a server and a server may perform the functions of a client.

[0020] Clients 110 may include client entities. An entity may be defined as a device, such as a wireless telephone, a personal computer, a personal digital assistant (PDA), a lap top, or another type of computation or communication device, a thread or process running on one of these devices, and/or an object executable by one of these device. Server 120 may include server entities that process, search, and/or maintain documents in a manner consistent with the principles of the invention. Clients 110 and server 120 may connect to network 140 via wired, wireless, or optical connections.

[0021] Clients 110 may include client software, such as browser software 115. Browser software 115 may include a web browser, such as the existing Microsoft Internet Explorer or Netscape Navigator browsers. For example, when network 140 is the Internet, clients 110 may navigate the web via browsers 115.

[0022] Server 120 may operate as a web server and include appropriate web server software 125. In one implementation, web server software 125 may function as a search engine, such as a query-based web page search engine. In general, in response to client requests, the search engine may return sets of

documents to clients 110. The documents may be returned to clients 110 as a web page containing a list of links to web pages that are relevant to the search query. This list of links may be ranked and displayed in an order based on the search engine's determination of relevance to the search query. Although server 120 is illustrated as a single entity, in practice, server 120 may be implemented as a number of server devices.

[0023] A document, as the term is used herein, is to be broadly interpreted to include any machine-readable and machine-storable work product. A document may be an email, a file, a combination of files, one or more files with embedded links to other files, a news group posting, a web advertisement, etc. In the context of the Internet, a common document is a web page. Web pages often include content and may include embedded information (such as meta information, hyperlinks, etc.) and/or embedded instructions (such as Javascript, etc.).

EXEMPLARY CLIENT/SERVER ARCHITECTURE

[0024] Fig. 2 is an exemplary diagram of a client 110 or server 120 according to an implementation consistent with the principles of the invention. Client/server 110/120 may include a bus 210, a processor 220, a main memory 230, a read only memory (ROM) 240, a storage device 250, one or more input devices 260, one or more output devices 270, and a communication interface 280. Bus 210 may include one or more conductors that permit communication among the components of client/server 110/120.

[0025] Processor 220 may include any type of conventional processor or microprocessor, or a combination of processors, that interprets and executes instructions. Main memory 230 may include a random access memory (RAM) or another type of dynamic storage device that stores information and instructions for execution by processor 220. ROM 240 may include a conventional ROM device or another type of static storage device that stores static information and instructions for use by processor 220. Storage device 250 may include a magnetic and/or optical recording medium and its corresponding drive.

[0026] Input device(s) 260 may include one or more conventional mechanisms that permit a user to input information to client/server 110/120, such as a keyboard, a mouse, a pen, voice recognition and/or biometric mechanisms, etc. Output device(s) 270 may include one or more conventional mechanisms that output information to the user, including a display, a printer, a speaker, etc. Communication interface 280 may include any transceiver-like mechanism that enables client/server 110/120 to communicate with other devices and/or systems. For example, communication interface 280 may include mechanisms for communicating with another device or system via a network, such as network 140.

[0027] The software instructions defining server software 125 and browser software 115 may be read into memory 230 from another computer-readable medium, such as data storage device 250, or from another device via communication interface 280. The software instructions contained in memory 230 causes processor 220 to perform processes that will be described later.

Alternatively, hardwired circuitry may be used in place of or in combination with software instructions to implement processes consistent with the principles of the invention. Thus, implementations consistent with the principles of the invention are not limited to any specific combination of hardware circuitry and software.

[0028] As mentioned, server software 125 may implement a search engine that, based on a user query, returns a list of links to documents that the server software 125 considers to be relevant to the search.

SERVER SOFTWARE 125

[0029] Fig. 3 is an exemplary functional block diagram illustrating an implementation of server software 125. Server software 125 may include a search component 305, a database component 310, and a spider component 315. Although search component 305, database component 310, and spider component 315 are illustrated in Fig. 3 as all being part of server software 125, one of ordinary skill in the art will recognize that these components could be implemented on separate computing devices or clusters of computing devices. Spider component 315 and search component 305, in particular, may be implemented independently of one another.

[0030] In general, search component 305 may receive user search queries from clients 110, search database 310 based on the search queries, and return a list of links (e.g., URLs) of relevant documents to clients 110. The list of links may also include information that generally attempts to describe the contents of the web documents associated with the links. The list of links may be ordered

based on ranking values, generated by search component 305, which rates the links based on their perceived relevance.

[0031] Database component 310 may store an index of the web documents that have been crawled by spider program 315. In one implementation, the index of the web documents may be an inverted index. Database component 310 may be updated as new web documents are crawled and added to database component 310. Database component 310 may be accessed by search component 305 when responding to user search queries.

[0032] Spider component 315 may crawl documents available through network 140. The content corresponding to the crawled documents may be forwarded to database component 310. Each crawled document may be identified by the URL that references the document. According to an aspect consistent with the principles of the invention, for dynamically generated URLs in which multiple URLs may refer to identical content, spider component 315 may generate and use rewrite rules to transform the URL into a canonical form. The canonical version of the URLs may ideally have a one-to-one correspondence with document content, potentially resulting in more efficient document crawling and document indexing. In practice, the canonical version of the URL generated by the rewrite rules may not have a perfect one-to-one correspondence with document content. That is, the rewrite rules may not perfectly reduce the many-to-one association of URLs to document content to a one-to-one relationship. However, just reducing the many-to-one relationship can produce advantages to server software 125.

[0033] Spider component 315 may include fetch bots 330, content manager 320, rewrite component 340, and URL manager 325. In general, fetch bots 330 may download content referenced by URLs. The URLs that are to be downloaded may be given to fetch bots 330 by URL manager 325, and may have been obtained from previously downloaded pages. URL manager 325 may keep track of the URLs that have been downloaded and what URLs are to be downloaded. Before URL manager 325 processes a downloaded URL in a page, the URL may be processed by rewrite component 340.

[0034] Content manager 320 may receive content downloaded by fetch bots 330 and destined for database component 310. Content manager 320 may process the content to extract URLs. URLs may be forwarded from content manager 320 to URL manager 325 and possibly to database component 310.

[0035] Rewrite component 340 may transform input URLs (e.g., URLs from content manager 320) into their canonical form (or an approximation of their canonical form). The transformation of input URLs to a canonical form may be based on a predetermined set of rewrite rules. A rewrite rule, may, for example, specify that when a URL is extracted from a particular web site with a particular parameter embedded in the URL, then the particular parameter should be removed from the URL.

[0036] Before describing generation of the rewrite rules for rewrite component 340 in additional detail, it may be helpful to examine the structure of an exemplary URL. Fig. 4 is a diagram illustrating an exemplary hypertext transfer

protocol (HTTP) URL 400. Other protocols, such as secure HTTP (HTTPs), may also be used.

[0037] Exemplary URL 400 begins with the HTTP prefix 401 indicating that the referenced content should be transferred using the well known hyper-text transfer protocol. Domain 402 specifies the particular entity or organization that is to handle the information request. In this example, the computing device (e.g., web server) "www" at the Internet address "bigonlinestore.com" is to handle the information request. Section 403 identifies the particular document that the computing device will return to the user. In this example, the document may be the active server page (".asp") "department" in the directory "retail."

[0038] Section 403 may additionally include a number of parameters used internally by the web server. Parameters may be separated from one another by the ampersand symbol ("&"). Thus, in the example, two parameters "loc" and "mp" are present. One of ordinary skill in the art will recognize that symbols other than the ampersand may be used to separate parameters. The parameter "loc" has the value 33410 and the parameter "mp" has the value 60. The parameters may generally be arbitrary parameters (e.g., random or non-random sequences of alpha-numeric characters) assigned by the web server, and may be used for a number of functions, such as assisting the web server in tracing user surfing patterns throughout the web site, tracking product categories, etc. Some of the parameters may have no affect on the delivered content.

[0039] Fig. 5 is a flow chart illustrating operations according to aspects consistent with the principles of the invention for generating rewrite rules for

rewrite component 340. Rewrite component 340 may begin by gathering URLs from a particular web site or web host (act 501). The URLs may, for example, be gathered from multiple web documents received by content manager 320 that were crawled by fetch bots 330. The crawl may be a "promiscuous" crawl that imposes few restrictions on dynamic URLs. The crawl may be biased toward novel parameter combinations in the crawled URLs. In general, the goal is to collect a number of URLs from a particular web site or web host that contain multiple parameter-value combinations that can be analyzed.

[0040] Rewrite component 340 may next choose a particular parameter combination that occurs in the URLs gathered in act 501 (act 502). The parameter combination may be a parameter combination that occurs a number of times in the gathered URLs. URL 400, for example, includes the parameter combination "loc" and "mp." Rewrite component 340 may then sample the gathered URLs to obtain at least a predetermined number (e.g., five) of different URLs that contain the chosen parameter combination (act 503). Fig. 6 is a diagram illustrating an exemplary sampled set of five URLs 601-605 gathered from the host "bigonlinestore.com" and that include the parameter combination "loc" and "mp." The values associated with the parameters in the sampled URLs 601-605 may be different between different samples 601-605.

[0041] Rewrite component 340 may then select and fetch the first URL (e.g., URL 601) from the sampled set and fetch the document, such as the web page, corresponding to the URL (acts 504 and 505). This corresponds to a fetch of the document with all the parameters. Rewrite component 340 may additionally

fetch the document with URL 601 having all the parameters removed, then each parameter individually, then all combinations of two parameters, and so on through various (or all) possible parameter combinations (act 506). For the exemplary URL 601, the possible URL parameter combinations in act 506 would include: "http://www.bigonlinestore.com/retail/department.asp?," "http://www.bigonlinestore.com/retail/department.asp?mp=60," and "http://www.bigonlinestore.com/retail/department.asp?loc=33410." The canonical version of URL 601 may then be selected as a URL (e.g., the first URL) from the combinations in act 506 that corresponds to approximately the same content as that of the document fetched in act 505 (act 507). By selecting the first URL in act 507, rewrite component 340 will select a URL with the least number of parameters but that corresponds to content that matches the original content. A document having "approximately the same content" as another document may be determined using any of a number of known document comparison techniques, such as comparison techniques based on a similarity hash. Approximate document content, instead of identical document content, may be useful because fetched documents are often generally identical but may contain certain elements, such as embedded advertisements, that may change.

[0042] Referring back to URL 601, assume that the URL with no parameters and the URL "http://www.bigonlinestore.com/retail/department.asp?mp=60," which includes the parameter "mp", returned a page different than that returned for the full URL 601 but that the page "http://www.bigonlinestore.com/retail/department.asp?loc=33410" returned

approximately the same content. In this situation, the parameter "loc" would be chosen as a parameter required in the canonical form of the URL.

[0043] Acts 504 through 507 may then be repeated for each additional URL 601-605 (act 508). In the example of Fig. 6, in which five URLs 601-605 were sampled, this would result in the generation of five URLs labeled as canonical URLs in act 507. If a parameter appears in a threshold number of these canonical URLs, such as in four or more, rewrite component 340 may label the parameter as required and generate a rewrite rule that rewrites a URL with the full set of parameters to one with only the required parameters (acts 509 and 510). The generated rewrite rule may then be used to generate canonical URLs from input URLs. For the example URLs given above, the rewrite rule may be: for each URL from the web host "bigonlinestore.com" that has the two parameters "loc" and "mp," remove the parameter "mp" and its associated value from the URL. Other rewrite rules are possible. For example, the rewrite rules may allow other unspecified parameters to be in the input URL. In other words, the URL could include a superset of the parameters in the rewrite rule. The rewrite rule may then either specify parameters to remove (any extra parameters remain in the rewritten URL) or parameters to keep (any extra parameters are deleted).The above example did not take the path into account (i.e., the string after the hostname and before the "?") when generating the rewrite rules. In alternate implementations, the path may be considered to be similar to a parameter and may be taken into account when generating rewrite rules.

CONCLUSION

[0044] As discussed above, rewrite rules are automatically generated that reduce the multiplicity of URLs that correspond to the same, or substantially similar, content. The rewrite rules may be applied when indexing, crawling, or otherwise processing content in order to reduce redundancy.

[0045] It will be apparent to one of ordinary skill in the art that aspects of the invention, as described above, may be implemented in many different forms of software, firmware, and hardware in the implementations illustrated in the figures. The actual software code or specialized control hardware used to implement aspects consistent with the present invention is not limiting of the present invention. Thus, the operation and behavior of the aspects were described without reference to the specific software code--it being understood that a person of ordinary skill in the art would be able to design software and control hardware to implement the aspects based on the description herein.

[0046] The foregoing description of preferred embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. For instance, some of the acts shown in Fig. 5 could be implemented in a different order or in parallel. Additionally, some of the acts may be omitted.

[0047] No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless

explicitly described as such. Also, as used herein, the article "a" is intended to include one or more items. Where only one item is intended, the term "one" or similar language is used.